NoSQL'e Giriş

NoSQL ve Big Data bir yapbozun parçaları gibidir, ikisi birbiriyle ilişkilidir.

Clientlerden gelen verileri vertical scaling (scale up) ile daha büyük bir depolama yerinde depolarken artık yetersiz kaldığı, depolanamayacak kadar çok büyük olan verileri parçalayarak horizontal scaling (scale out) birden fazla parça halinde depolanır. Bunu açıklamak gerekecek olursa 1TB'lık bir harddisk almak yerine 10TB'lık bir harddisk alarak fazladan maliyet oluşturmak yerine var olan 1TB'lık harddiskten ihtiyacımız kadar doldukça yenisini ekleyerek hem maliyetten hem de oluşabilecek teknoloji açıklarından zarar görmeyiz.

Scalability = Ölçeklenebilirlik

Horizontal Scale Out = Yatay Ölçeklendirme

Vertical Scale Out = Dikey Ölçeklendirme

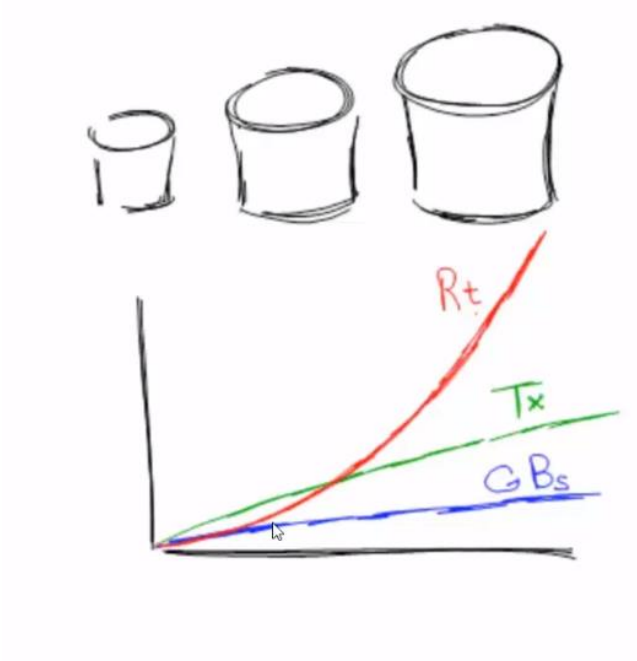
Veri Çeşitliliği – Data Variety

Structered – Yapısal

Semi – Structered – Yarı Yapısal

Unstructered – Yapısal Olmayan

Query Response Time – Sorgu Cevaplama Süresi



Mavi = Veri Büyüklüğü

Yeşil = Query Sayısı

Kırmızı = Sorgu Süresi

Verinin büyüklüğüne bağlı olarak sorgu atan kişi sayısı fazlaysa eksponansiyel bir artış olur.

RDMS ACID	NoSQL BASE
Atomicity: Bütünlük, ya hep ya hiç Consistency: Tutarlılık, mevcut verinin yeni değeri alabilmesi. Isolation: Tek işlem, ilgili veri kilitlenir işlemler seri halde yapılır. Durability: Dayanıklılık, hataya dayanıklılıktır. Hata anında eski değere dönebilir. Dikey Ölçeklendirme Scale Up	Basic Availability: Amaç hızlı cevap ve erişilebilir olmaktır. Soft State: Tutarlılık konusunda esnektir. Eventualy Consistency: İşlemler sistemin durumuna göre diğer sunuculara yansır, neredeyse tutarlı bir sistemdir. Yatay Ölçeklendirme Scale Out

ACID vs BASE

Tutarlılık vs Erişilebilirlik

Dikey Ölçeklendirme: Kutunun içerisine sığmayan verileri sığdırmak için kutuyu büyütürüz.

Yatay Ölçeklendirme: Kutunun içerisine sığmayan veriler için yeni bir kutu alırız.

CAP Teoremi

Dağıtık bir sistemde veri üzerinden sunulan hizmet için aynı anda 3 özellik sağlanamaz!

- Consistency (Tutarlılık): DS'de bir sunucuya Kullanıcı Adı: DS yazdınız. Başka bir sunucuya Kullanıcı Adı = ? sorgusunu arattığınızda DS cevabını aldınız. Bu duruma tutarlılık denir.
- Availability (Erişilebilirlik): Her zaman erişebildiğiniz bir sistem. Hem yazma hem okumada her zaman cevap alabildiğiniz bir cluster (küme). Bu duruma erişilebililirlik denir.
- Partition Tolerance (Bölünebilme Toleransı): Cluster da sunucular arasında bağlantı gittiğinde de çalışmaya devam edebilir. Bu duruma bölünebilme toleransı denir.
- Consistency U Availabtility = CA
- Consistency U Partition Tolerance = CP
- Avilability U Partition TOlerance = AP

CA'da Neden P Olmuyor ?

- Clusterdaki sunucularımızda ağ bağlantı sorunu yaşanmış. Yani sunucularımız arasındaki bağ çözülmüş. Bir veri geldi ve bir numaralı sunucuya yazıldı. Diğer sunuculara da veriyi yazması gerekiyor çünkü Consistency istiyoruz fakat yazamıyor bağlantı yok. Diğer sunuculara sorgu gönderdiğimizde veriyi bulamayacak. Hem verinin tutarlı olması hem de her zaman erişilebilir olmasını istediğimizde Bölme Toleransını sağlayamıyoruz.

AP’de Neden C Olmuyor ?

- 3 sunuculu bir clusterda tüm sunucularda Kullanıcı Adı: DS verisi bulunuyor. Yine sunucular arasında ağ bağlantı sorunu yaşanıyor. 1. Sunucuda Kullanıcı Adı: DSAdmin olarak güncellendi. 2. Sunucu üzerinde sorgu attığımızda bağlantı kaybolduğu için orada eski veri var. Bize sonuç Kullanıcı Adı: DS dönecektir. Bu durumda Consistency olmadı. Sosyal Medya liderleri platformun içerisinde tutarlılığı önemsemezler.

CP’de Neden A Olmuyor ?

- Sistemin hem tutarlı olmasını istiyoruz hem de verinin network üzerinde dağıtılmasını istiyoruz fakat cluster da bir sunucuda bağlantı koptuğu an erişilebilirlik ortadan kalkacak. Kopan sunucuyu kapatmalısınız yoksa tutarlılık bozulacaktır. MongoDB veri tabanı gibi sistemler CP uygundur. Defaultta Strongly Consistent’dır.

Büyük Veri Nedir ?

- Günümüzde son derece değerli ve popüler bir teknoloji olan Büyük Veri, teknolojinin ilerlemesiyle birlikte gün geçtikçe daha çok değerleniyor. Dünyada birçok şirket, veri ile birlikte dijital dönüşümü gerçekleştirirken, bu dönüşüm Büyük Veriyi yeni bir devrin başlangıcı olarak tanımlıyor.
- Büyük Veri 2000’li yıllarında başında analist Doug Laney ile popülaritesinde ivme kazanmaya başlamıştır. Büyük Veri kavramı büyük hacimli yapılandırılmış ve yapılandırılmamış verileri ifade etmek için kullanılmaktadır. Bu veriler o kadar büyük, hızlı ve komplikedir ki geleneksel yöntemler kullanarak işlemek mümkün değildir.

NoSQL Çeşitleri

- Document: Couchbase, RAVENDB, MongoDB, MarkLogic, CouchDB, Cloudant
- Key-Value: amazon DynamoDB, redis, Coucbase, riak
- Wide-Column: Scylla, cassandra, Azure Cosmos DB, Apache HBASE
- Graph: OrientDB, Arango, neo4j

Doküman Veri Tabanları

- Doküman tabanlı NoSQL veri tabanları, verileri JSON benzeri dökümanlar olarak depolamak ve sorgulamak için tasarlanmış, ilişkisel olmayan bir veri tabanı türüdür.
- Döküman veri tabanları yazılımcıların kodlamada kullandıkları aynı doküman model biçimini kullanarak verileri bir veri tabanında depolanmasını ve sorgulanmasını kolaylaştırır.
- Veriler esnek, yarı yapılandırılmış ve hiyerarşik bir biçimde tutulur.

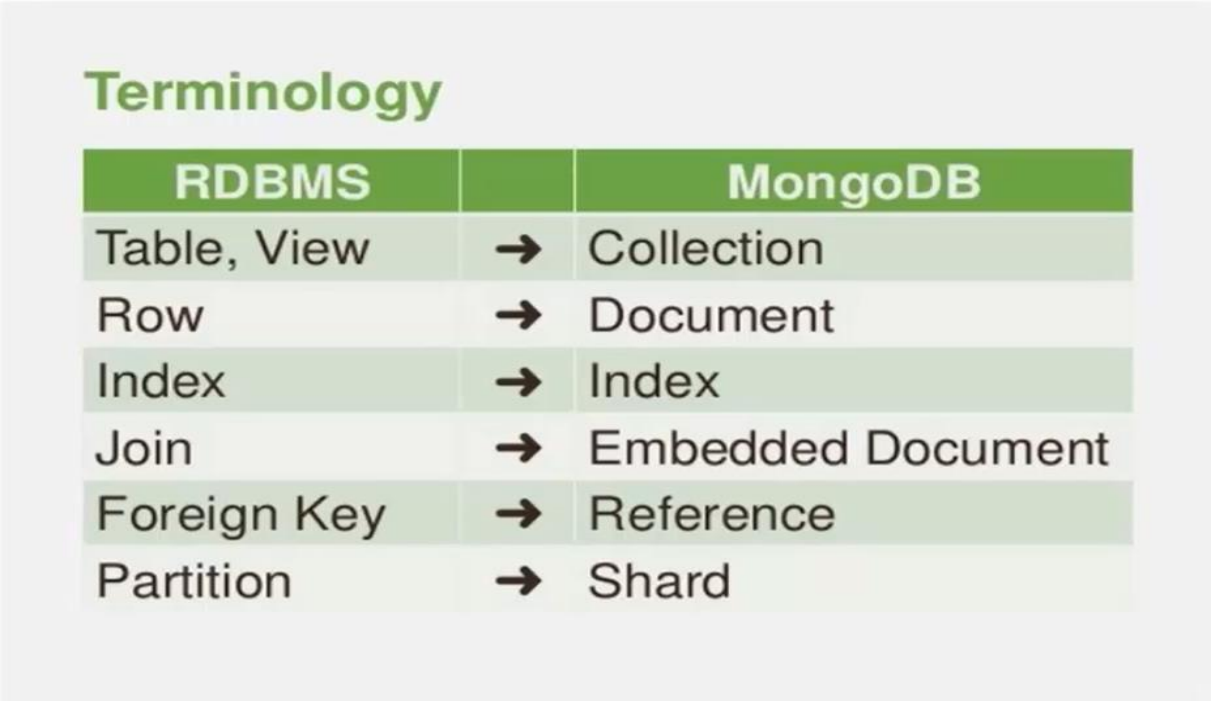
İlişkisel Veri Tabanlarında Doküman Kullanabilir Miyiz ?

- İlişkisel veri tabanlarından bir kısmı mimarilerine JSON desteği eklenmiştir.
- Döküman veri tabanları yazılımcıların kodlamada kullandıkları aynı doküman model biçimini kullanarak verileri bir veri tabanında depolanmasını ve sorgulanmasını kolaylaştırır.
- Veriler esnek, yarı yapılandırılmış ve hiyerarşik bir biçimde tutulur.

2. MongoDB Temelleri

- Belge tabanlı (document-oriented) bir NoSQL veri tabanıdır.
- mongoDB'de her kayıt bir doküman olarak tutulur.
- Dökümanlar JSON benzeri binary JSON (BSN) formatında saklanır.

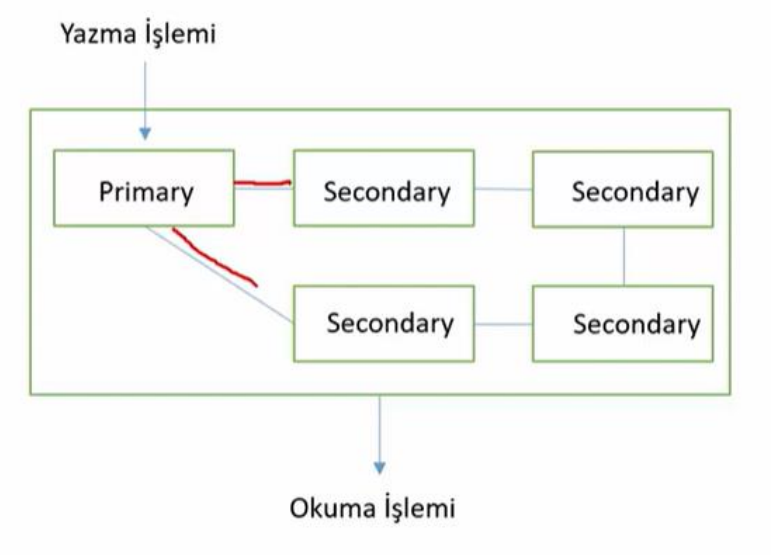
Terminology



mongoDB Özellikleri

- Açık Kaynak
- Doküman Tabanlı
- Schema Bağımsız Olması
- Scalability (Ölçeklenebilirlik)
- Replikasyon (Kopyalama)
- Load Balancing (Yük Dengeleme)
- Sorgulama Kolaylığı
- Indexleme

mongoDB Cluster Replication Sets

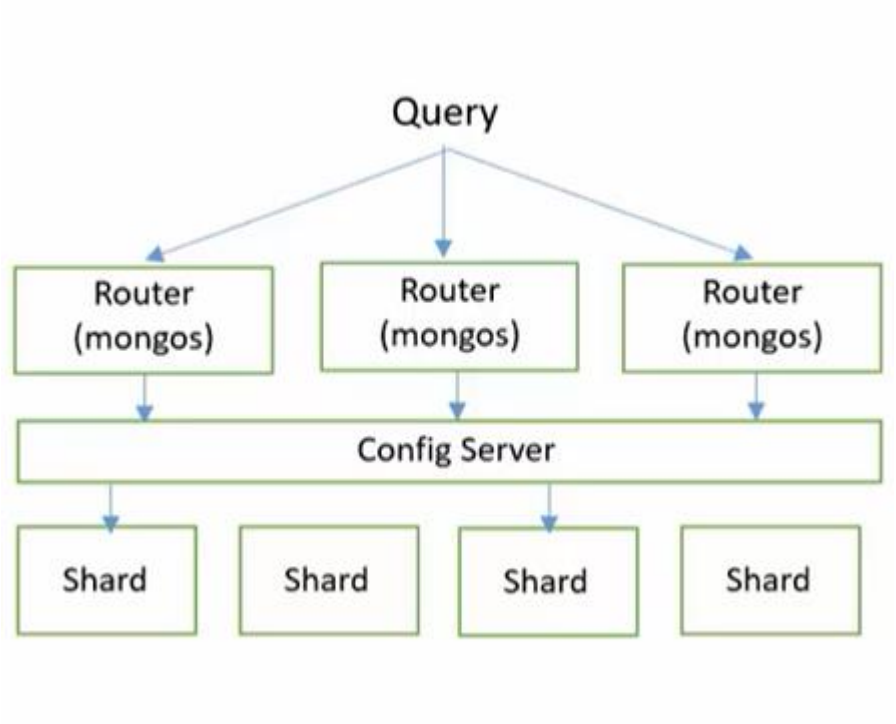


Tek Lider (Master, Primary)

Primary Node'a yaz.

Secondary Node'lardan oku.

mongoDB Cluster Sharding



Büyük Veriler Kontrol Edilmeli

Verileri Belirli Bir Alana Göre Parçalara Ayırmalıyız

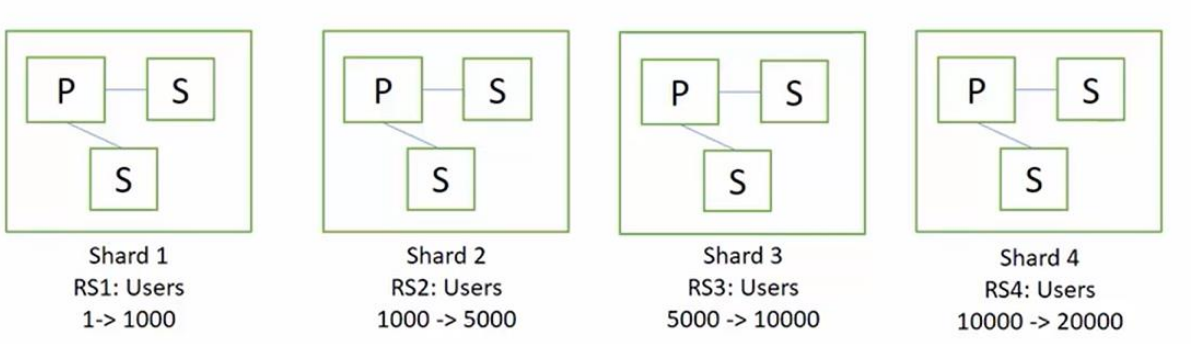
Mongos, Config Server, Shards

Mongos: Router görevindedir. Queryleri yönlendirir ve clientlara veri dönmeden shardlardan aldığı verileri birleştirir.

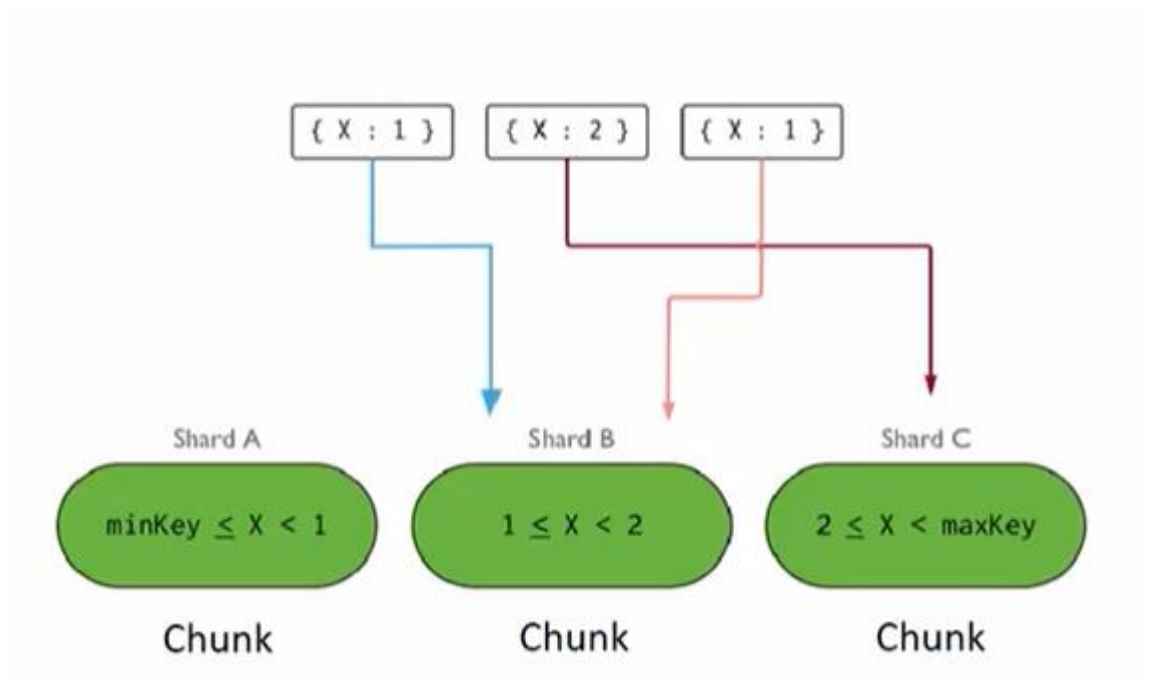
Config Server: Metadata, shard ve cunk bilgisi tutulur.

Shard: Verileri tutan bölümdür. Mongos'un yönlendirilmesi ile veriler key sayesinde shardlara bölünür ve chunklar şeklinde tutulur.

mongoDB Cluster Sharding – Verileri Bölmek



mongoDB Cluster Sharding – Shard Key



Shard Key – Shard: Veriler, Key: İstenilen Veri

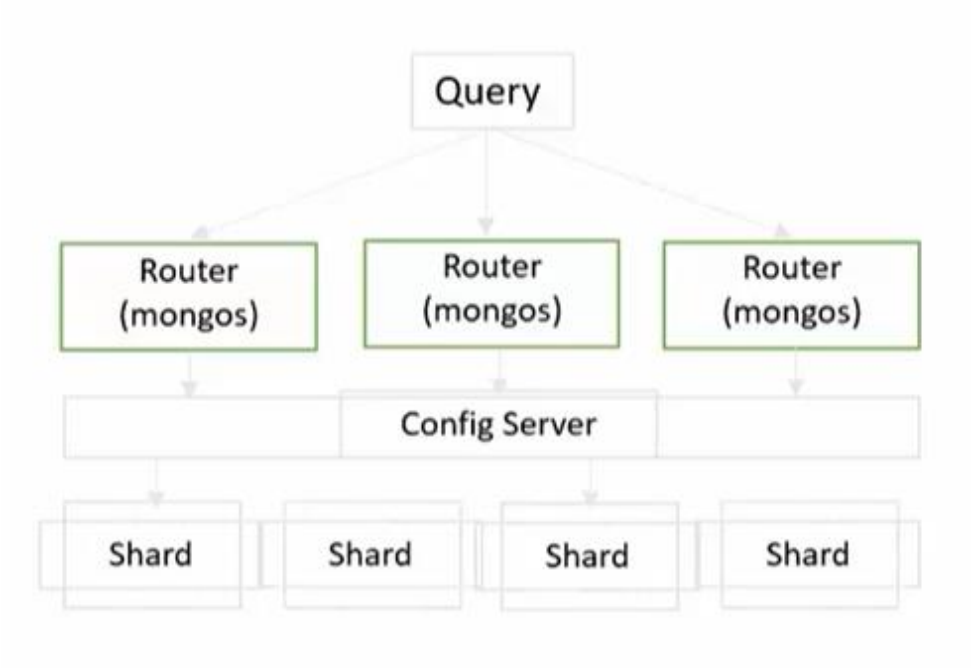
- 10 yıllık fotoğraf arşivinden sadece 2015 yılına ait fotoğrafları arıyoruz. Bunun için bilmemiz gereken fotoğrafın hangi yılda çekildiği.

- 1 Shard bir veya birden fazla aralıktan sorumlu olabilir.
- Doğru field (alan) seçilmelidir.
- Yanlış seçimler ileride başımızı ağrıtabilir.
- Devamlı artan bir Shard Key iyi değildir.
- Rastgele değerlere sahip Shard Key iyi değildir.
- Forsquare 2010 yılındaki 17 saatlik kesintisi iyi bir örnektir.

mongoDB Cluster Sharding - Chunk

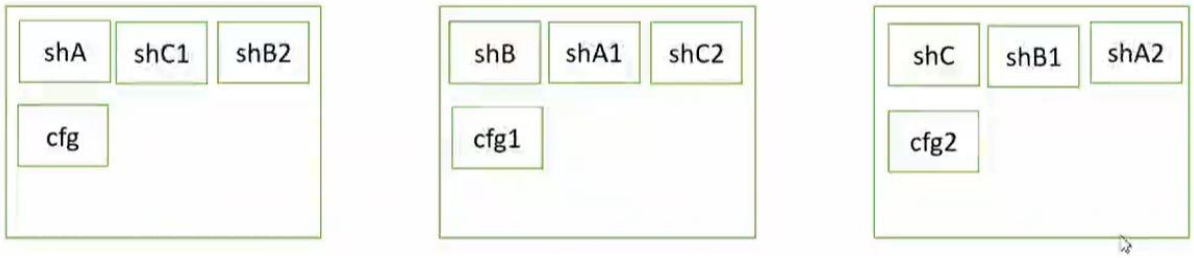
Bir shardda birden çok chunk bulunabilir ama bir chunk birden çok shardda bulunamaz.

mongoDB Cluster Sharding – Mongos



- Kullanıcı (Admin) ile Cluster arasında iletişim sağlar.
- Tüm okuma ve yazma mongosdan başlar.
- Query'de belirtmediğimiz sürece shardlara direkt ulaşım olmaz.
- Direkt ulaşım için query içerisinde shardkey kullanılmalıdır.

mongoDB Cluster Mimarisi Replica Set

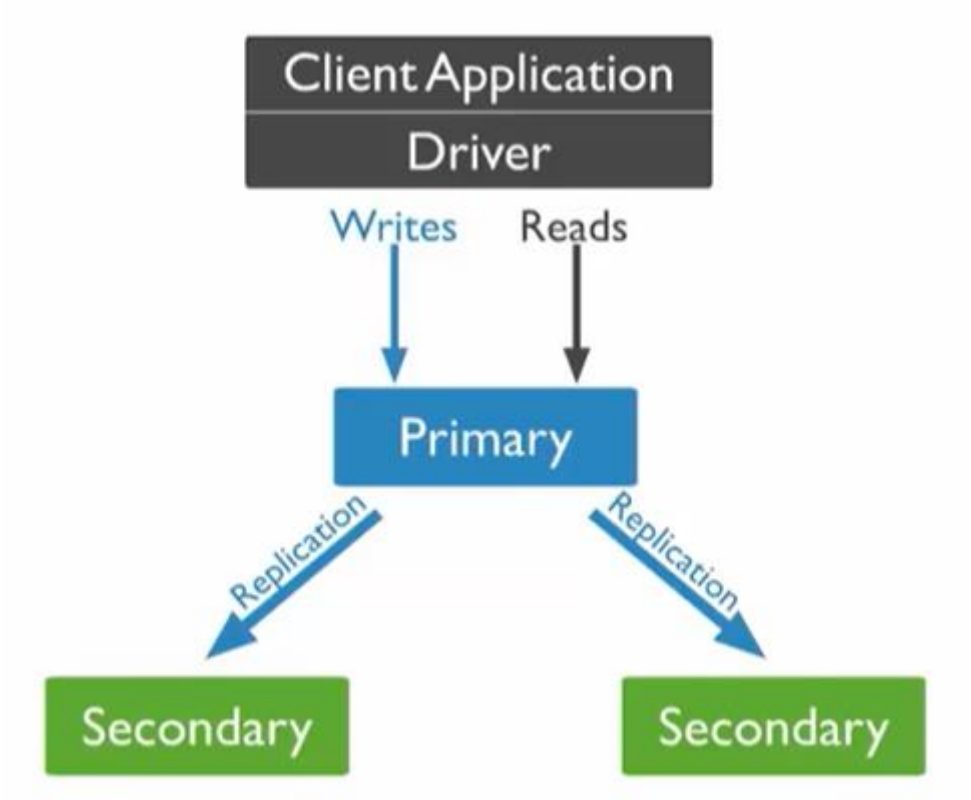


3 Server (Node) – Replica Set - Sharding – 3 Shard – shA, shB, shC

mongoDB Cluster Problemleri

Elde olmayan sebeplerden dolayı network kopması sonucunda cluster iki parçaya ayrılır. İki parçaya ayrılan clusterların her biri kendileri için Primary Node belirlerler. Bir süre sonra network bağlantısı yeniden sağlandığında tek bir cluster haline geçiş yapılmasına Split Brain denir. (Tek sayıda cluster bulundurarak problemin önüne geçilebilir.) Yine network kopması sonucunda ikiye ayrılan clusterların sayısı ne tarafta fazla ise o tarafın dediğini olur.

Node Çeşitleri

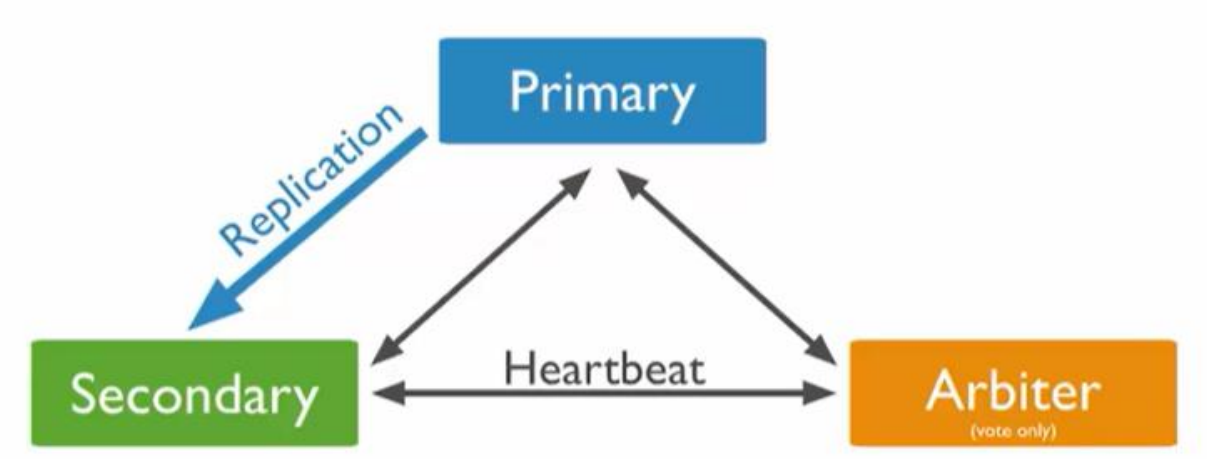


Primary Node:

- ReplicaSet'de yazma işlemini alabilen tek nodedir.
- Yazma işlemlerini Oplog'a kayıt eder.
- Diğer nodelar Oplog'dan veriyi çeker. (OpLog = Operation Log)
- Bir ReplicaSet'de en fazla 1 tane PRIMARY Node olabilir.
- PRIMARY Node kullanılmazsa kalan üyeler arasından yeni PRIMARY Node oy ile seçilir.

Secondary Node:

- ReplicaSet'de bir ya da birden fazla bulunabilir.
- Secondary Nodelar Oplog'dan veriyi çeker.
- Sadece okuma için kullanılır.



Arbiter Node

- ReplicaSet'de bulunması zorunlu değildir.
- Amacı oy vermektir.
- Birden fazla Secondary varsa gerekmemektedir.
- PSA(Primary – Secondary – Arbiter) Mimaride kullanılır.
- Veri tutmaz.
- Çok düşük kaynak kullanır.

mongoDB Shell

5'in üzerindeki versiyonlarda mongo yazdığınız zaman hata verecektir. BTK'da olan dersle senkron gidebilmek için 5 versiyonunu kurdum. Mongoyu kullanabilmek için yapmanız gereken birkaç adım var.

Dosya Yolu: C:\Program Files\MongoDB\Server\5.0\bin

Bilgisayarıma sağ tıklayıp Özellikler dedikten sonra açılan pencerede Gelişmiş Sistem Ayarlarına tıklayın. Açılan pencerenin altında Ortam Değişkenlerine tıklayın. Alt kısımda bulunan Path kısmına çift tıklayarak açın ve içerisine Dosya Yolunu yapıştırıp, tüm sekmeleri tamam diyerek kapatın. Terminali tekrardan açıp mongo yazınca çalışacaktır.

show dbs = Daha önce oluşturulmuş veri tabanlarını gösterir.

use dbname = Kullanmak istediğimiz veri tabanını seçeriz.

Collection = MySQL'de ki tabloya eş değerdir.

show collection = Koleksiyonları listeler.

CREATE

```
db.createCollection("intern") = Intern koleksiyonunu oluşturur.
```

find = SQL'de ki SELECT fonksiyonuna eş değerdir.

INSERT

```
db.intern.insert{"name": "Dogukan","surname": "Bostancı","age": 21,"city":
"Bursa","phone":[{"type": "home", "number": 2518629}, {"type": "mobile", "number": 5431897471
}]}
```

FIND

```
db.intern.find({“name”:”Dogukan”}) = (key = value) değer araması yapılır.
db.intern.find({age:{$gte:20}}) = 20 yaşına eşit ve büyük verileri getirir.
db.intern.find({age:{$lte:20}}) = 20 yaşına eşit ve küçük verileri getirir.
db.intern.find({age:{$gt:20}}) = 20 yaşından büyük verileri getirir.
db.intern.find({age:{$lt:20}}) = 20 yaşından küçük verileri getirir.
```

UPDATE

```
db.intern.update({"name":"Dogukan", "surname":"Bostancı"}, {"name":"Doğukan", "surname":"Bostancı",
"age": 21,"city":"Bursa","phone":[{"type": "home", "number": 2518629}, {"type": "mobile", "number":
5431897471 }]})
```

REMOVE

```
db.intern.remove({"name":"Doğukan"}) = Belirtilen değeri siler.
```

Query Syntax

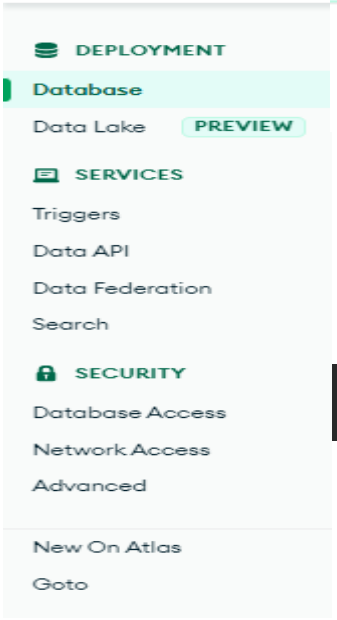
Soru1: İsmi Doğukan olan kullanıcıları listeleyen sorguyu yazınız.

```
{ "name" : "Doğukan" } – Studio 3T Query
```

MongoDB Query Code

```
// Requires official MongoShell 3.6+

db = db.getSiblingDB("TrexGo");

db.getCollection("intern").find(

{

"name" : "Doğukan"

}

);
```

MongoDB Atlas IP = 0.0.0.0/0 => Tüm Dünya üzerinden bağlantı sağlanabilir.



- Database kısmı seçildikten sonra gelen ekranda Browse Collection'a tıklanır.
- Load Sample Data seçeneğinden rastgele bir dataset indirilir.
- Atlasta bulunan rastgele database Studio 3T'ye bağlanr.
- Studio 3T'de bağlantı yaparken <password> yazan kısım silinir ve öncesinde belirlediğiniz şifre yazılır.

Bu kısımdan sonra yapılacak tüm sorgular sample_restaurant database içerisinde yer alan restaurant koleksiyonunda gerçekleşecektir.

Studio 3T'de üst kısımda bulunan **Aggregate** kısmına;

Pipeline sekmesi -> Add Stage -> Operator: $group adımları izlenir.

Toplam mutfak sayısını veren sorguyu yazınız.

```
_id: "$cuisine", "toplam": {$sum:1}
```

Mutfak sayısını çoktan aza sıralayan sorguyu yazınız.

```
"toplam": -1
```

En çok hangi ilçede mutfak olduğunu veren sorguyu yazınız.

```
_id: "$borough", "toplam": {$sum:1}
```

User – Role İşlemleri

Studio 3T'de üst tarafta bulunan User sekmesine tıklandığı zaman okuma/yazma hatası verecektir. Bunu düzeltmek için Atlasta sol panelde bulunan Database Access sekmesine girilir. Açılan ekranda sağ üstte bulunan Add New Database User kısmından yeni kullanıcı eklenir. Bu kısımda dikkat etmeniz gereken nokta ekleyeceğiniz kullanıcının database üzerinde nasıl bir yetkisi olacağıdır. Read, write, cluster monitör, dbadmin gibi seçeneklerden birisini Specific Privileges kısmından seçmelisiniz.

Yeni eklediğimiz kullanıcıyı test etmek için New Connection kısmına gelip atlas sunucumuzu seçip Edit diyoruz. Açılan pencerede eklediğimiz kullanıcı adını ve şifresini girip sol altta bulunan Test Connection'a tıklayıp herhangi bir sorun olup olmadığını kontrol ediyoruz.

Custom Role kısmından kişilerin çalıştıkları departmanlara ve iş tanımlarına göre yetkilendirme yapılabilir.

Data API

- Öncelikle Postman kurun ve daha sonrasında adımları takip edin.
- Sol panelde yer alan Data API paneline girin.
- Açılan sayfada sağ üstte bulunan Test Your API yazan butona tıklayın.
- API'ye isim verip data kaynağını, database ve koleksiyonları seçin.
- Aşağıda bulunan cURL alanını kopyalayın ve Postman'i açın.

- Postman dosya kısmından import seçeneğini seçin ve kopyaladığınız cURL'yi yapıştırın. Body kısmına gelip projection kısmını silip send butonuna basın.
- POST kısmında çıkan cURL'nin sonunda bulunan findOne ifadesindeki One ifadesini silin.One ifadesini silmezseniz sadece 1 veriyi listeleyecektir.
- Body içerisine "filter": {"cuisine": "Turkish"} sorgusunu yazıp send butonuna tıkladığınızda sorgu çalışacaktır.